Project Reports: AI Dungeon Story Generator & Face Mask Detection with Live Alert System

AI Dungeon Story Generator

Introduction: The AI Dungeon Story Generator is an interactive storytelling platform that uses generative AI models to create fantasy or mystery stories. It allows users to choose a genre, enter a prompt, and generate unique continuations powered by pretrained GPT models.

Abstract: This project uses GPT-based language models such as GPT-2 or GPT-Neo to generate dynamic storylines based on user input. By leveraging Streamlit for the frontend, the application provides an engaging and interactive storytelling experience where users can explore AI-generated adventures.

Tools Used: Python, Hugging Face Transformers, Streamlit, Text File Handling.

Steps Involved: 1) Loaded pretrained GPT-2/GPT-Neo model from Hugging Face. 2) Created a Streamlit interface for user input and genre selection. 3) Implemented prompt-based story continuation logic. 4) Generated multiple possible story continuations. 5) Enabled saving the final story as a text file. 6) Deployed the app using Streamlit sharing or local hosting.

Conclusion: The AI Dungeon Story Generator demonstrates how generative AI can be used for creative storytelling, offering personalized and imaginative experiences. Future improvements could include memory-based story progression and voice-based narration.

Face Mask Detection with Live Alert System

Introduction: This project aims to detect whether people are wearing face masks in real-time using a webcam feed. It combines classical computer vision techniques for face detection with a Convolutional Neural Network (CNN) classifier to determine mask presence and triggers live alerts when masks are not detected.

Abstract: The Face Mask Detection system uses a trained Keras CNN model to classify faces as masked or unmasked. Haar Cascade classifiers (or modern alternatives like MTCNN) detect face regions in live video frames captured via OpenCV. When an unmasked face is detected, the system issues configurable alerts (sound, on-screen warning, and optional email/SMS via Flask backend). The project includes dataset preparation, model training, real-time integration, and deployment options.

Tools Used:
• Python (OpenCV, Flask) • TensorFlow / Keras • Haar Cascades (cv2) or MTCNN for face detection • Kaggle dataset (masked / unmasked faces) or custom dataset • Supporting libraries: NumPy, imutils, scikit-learn

Steps Involved in Building the Project: 1. Dataset: Collect or download a labeled dataset of masked and unmasked faces (e.g., Kaggle). Split into train/validation/test sets and apply augmentation (rotation, zoom, flips). 2. Preprocessing: Resize face ROIs (e.g., 128x128), convert to RGB or grayscale as required, normalize pixel values, and one-hot encode labels. 3. Model: Build a lightweight CNN in Keras (Conv -> ReLU -> Pooling -> Dropout -> Dense). Compile with Adam optimizer and categorical_crossentropy (or binary_crossentropy). 4. Training: Train the model with callbacks (EarlyStopping, ModelCheckpoint) and evaluate performance on validation set. Save the final model (HDF5 or SavedModel format). 5. Face Detection Integration: Use OpenCV Haar Cascade to detect faces in webcam frames. Extract ROI, preprocess, and pass to the trained model for classification. 6. Alert Logic: If prediction == 'No Mask' with confidence > threshold, trigger alerts: play alarm sound, overlay red bounding box and warning text, increment counter. Optional: send HTTP request to Flask endpoint to log events or send notifications (email/SMS) via SMTP or third-party services. 7. Deployment: Create a simple Flask app to serve status, logs, and optionally stream processed video. Containerize with Docker for production deployment. Prepare a short demo video showing real-time detection and alerts.

Deliverables: Trained model file (e.g., mask_detector.h5), real-time detection script (mask_detector_live.py), optional Flask deployment (app.py), short demo video (MP4), and a GitHub repository with code and instructions.

Conclusion: The project provides a practical real-time solution for mask compliance monitoring using accessible tools. It can be extended with better face detectors, multi-camera support, cloud alerts, and dashboard analytics.